Realizar una conversion de Java List to Map es bastante habitual. La mayor parte de las veces cuando trabajamos con Java y con clases de Repositorio y Servicios devolvemos listas de objetos. Sin embargo a veces nos puede ser muy interesante realizar una conversión a mapa. Vamos a ver un ejemplo sencillo de estas cosas. Para ello partiremos del concepto de ModeloCoche.

```
package com.arquitecturajava;

public class ModeloCoche {

  private String identificador;
  private String marca;
  private double precio;
  public String getIdentificador() {
    return identificador;
  }
  public void setIdentificador(String identificador) {
    this.identificador = identificador;
  }
  public String getMarca() {
    return marca;
  }
  public void setMarca(String marca) {
    this.marca = marca;
  }
  public double getPrecio() {
    return precio;
  }
  public void setPrecio(double precio) {
    this.precio = precio;
  }
```

```
  public ModeloCoche(String identificador, String marca, double
precio) {
    super();
    this.identificador = identificador;
    this.marca = marca;
    this.precio = precio;
  }

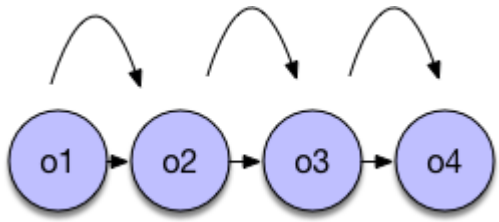
}
```

Vamos a construir una lista de Modelos Coche:

```
 ModeloCoche modelo1 = new ModeloCoche("Y1Yaris", "Toyota", 16000);
 ModeloCoche modelo2 = new ModeloCoche("Y2Yaris", "Toyota", 20000);
 ModeloCoche modelo3 = new ModeloCoche("Y3Yaris", "Toyota", 22000);
 List < ModeloCoche > modelos = Arrays.asList(modelo1, modelo2,
modelo3);
```

Java List to Map

En estos casos tenemos una lista de elementos pequeña.

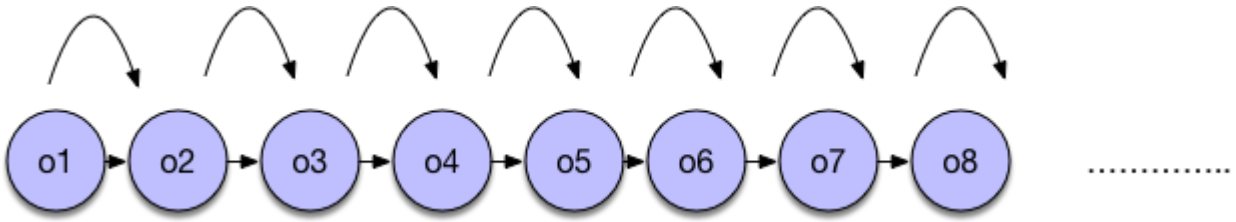


Supongamos que deseamos buscar dentro de la lista el Toyota Yaris “Y1Yaris” e imprimir sus datos por pantalla. Con Java 8 es relativamente sencillo. Podemos usar un stream y el método findFirst que nos devuelve un optional.

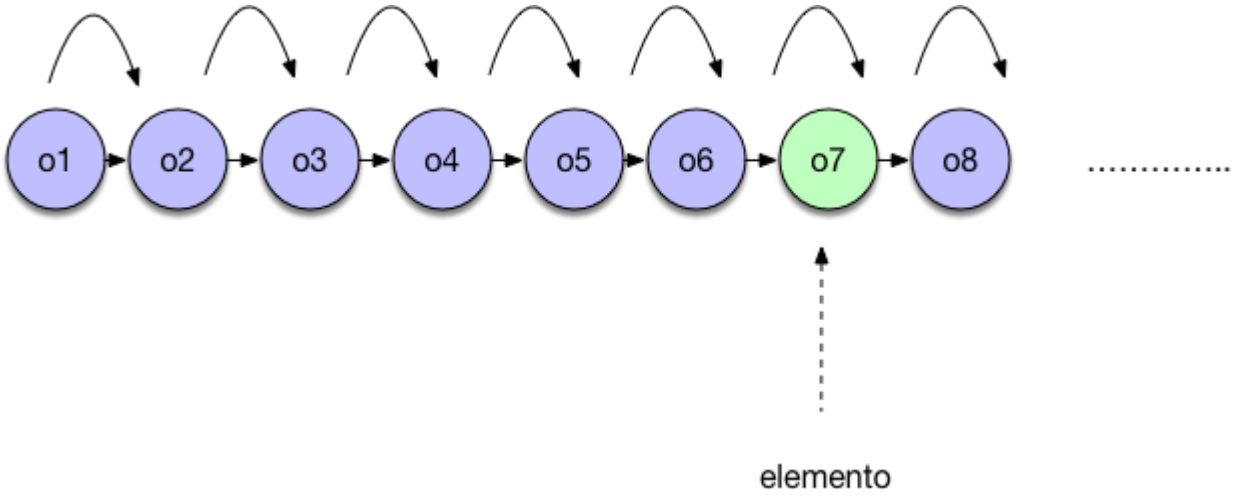
```
Optional < ModeloCoche > optional = modelos.stream().filter(m ->
m.getIdentificador().equals("Y1Yaris")).findFirst();
```

```
if (optional.isPresent()) {
  System.out.println(optional.get().getMarca());
  System.out.println(optional.get().getPrecio());
```

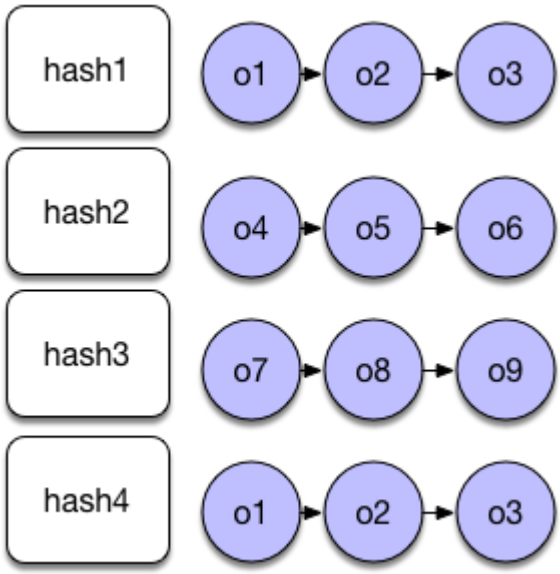
¿Es esta la solución correcta? . La realidad es que “depende” . Depende de si sobre esta lista vamos a buscar una única vez o vamos a buscar muchas veces. Por ejemplo imaginémonos que la lista de elementos contiene todas las marcas y los modelos de los coches. Puede ser una lista muy grande de elementos que nos interese tener cargada en memoria en una aplicación web ya sea a través de un mecanismo de cache o a través del ServletContext.



Si la consultamos en muchas páginas para sacar la información de uno u otro modelo de coche. Tendremos continuamente que buscar dentro de la lista un único resultado.



Sería mucho mejor almacenar esta estructura en memoria convertida en un mapa. Los mapas nos permiten acceder mucho más rápido a un elemento a través de su hash ya que accedemos a una sublista.



¿Cómo podemos hacer esto? . Muy sencillo podemos usar Java 8 y el collector de mapas para convertir de forma directa la estructura de list a una estructura map

```
        List < ModeloCoche > modelos = Arrays.asList(modelo1, modelo2, modelo3);

        Map < String, ModeloCoche > mapa = modelos.stream().collect(Collectors.toMap(x -> x.getIdentificador(), x -> x));

        ModeloCoche nuevoModelo1 = mapa.get("Y1Yaris");

        System.out.println(nuevoModelo1.getMarca());
        System.out.println(nuevoModelo1.getPrecio());
```

De esta forma será mucho más rápido y mucho más sencillo acceder a los datos de un elemento concreto a través de del uso de su identificador. Acabamos de usar Java Streams para realizar una conversión automática de Java List to Map. Este tipo de colectores nos puede ser muy útil cuando necesitemos almacenar información que va a ser continuamente consultada a traves de una cache o mecanismo similar.

Otros artículos relacionados

- Java Stream Collectors y su uso
- ¿Que es un UUID?
- Java Array to Stream y sus opciones
- Java Stream to List y simplificación